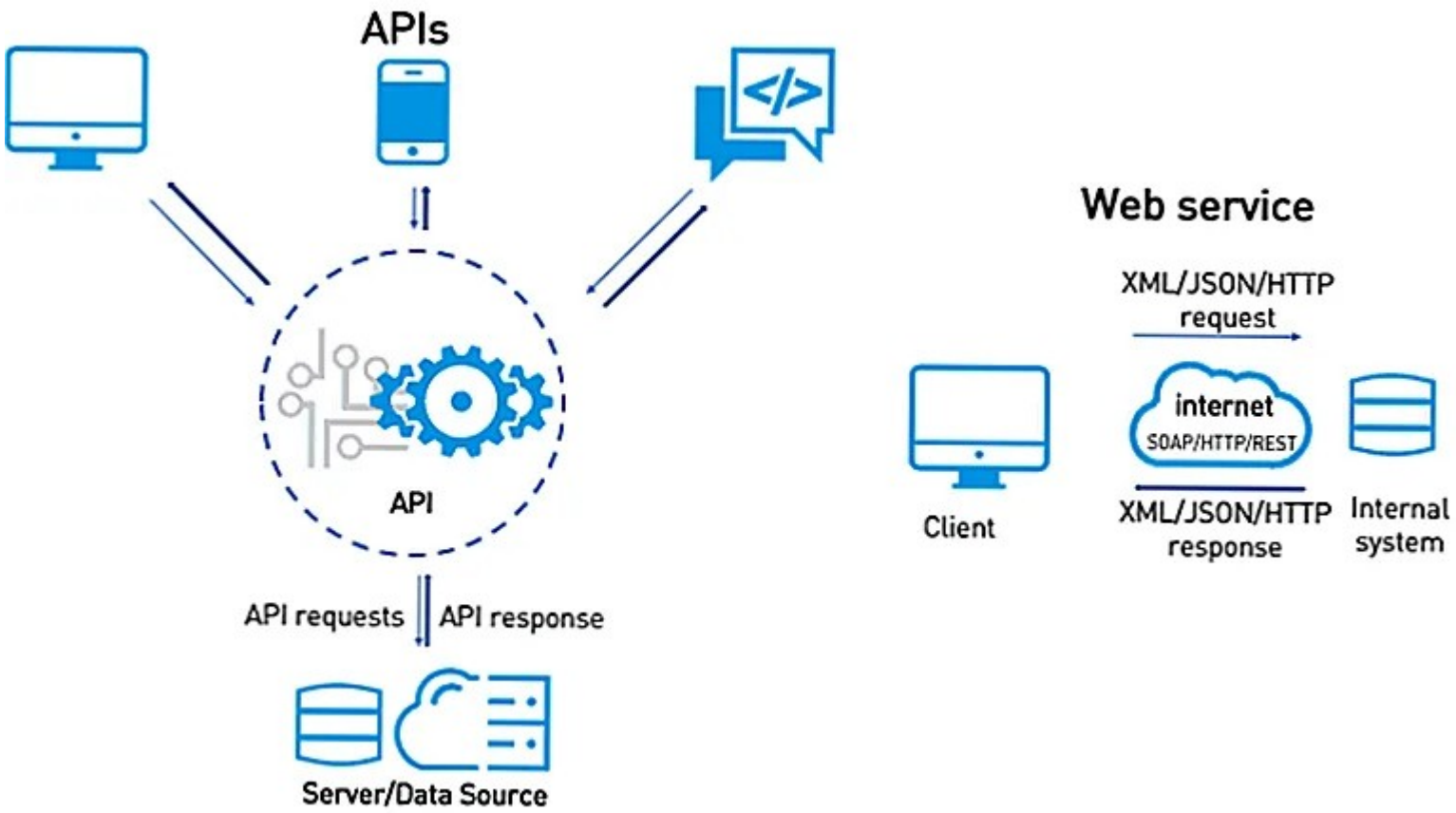
Micro framework Flask

Web services
REST & RESTful

Web service

- Web service – сервис (набор методов) , предоставляемый приложением и доступный по сети
- Стандартизированный способ взаимодействия разнородных приложений
- Предаставление услуг для любого приложения


APIs
API
API requests
API response
Server/Data Source
Web service
XML/JSON/HTTP
request
internet
SOAP/HTTP/REST
XML/JSON/HTTP
response
Client
Internal
system

Что такое REST ?

- Технологию REST впервые представил один из авторов HTTP-протокола Рой Филдинг (Roy Fielding) в своей диссертации в Калифорнийском университете в Ирвайне (2000 год).
- REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем.
- REST - **НЕ** протокол
- REST является простым интерфейсом управления информацией
- Системы, поддерживающие REST, называются RESTful-системами

Принципы проектирования RESTful Web-сервисов

- Явное использование HTTP-методов.
- Взаимодействие между сервером и клиентом не хранит состояние (stateless).
- Предоставление URI, аналогичных структуре каталогов.
- Передача данных в XML, JavaScript Object Notation (JSON).

Какие методы HTTP задействует REST

SQL(CRUD)	REST(HTTP)
SELECT	GET
INSERT	POST
UPDATE	PUT
DELETE	DELETE

REST(URI) API design

URI	HTTP-метод	Действие
/student/list	GET	получить список всех студентов
/student/{id}	GET	получить студента по id
/student/{id}	POST	изменить студента (данные в теле запроса)
/student/	PUT	добавить студента (данные в теле запроса)
/student/{id}	DELETE	удалить студента

Выбор между REST или SOAP

SOAP веб-сервисы

- XML, WSDL
- Работа с методами
- Поддержка транзакций, уровней безопасности и пр.
- Большое кол-во спецификаций
- Различные транспортные уровни
- Сложнее в разработке

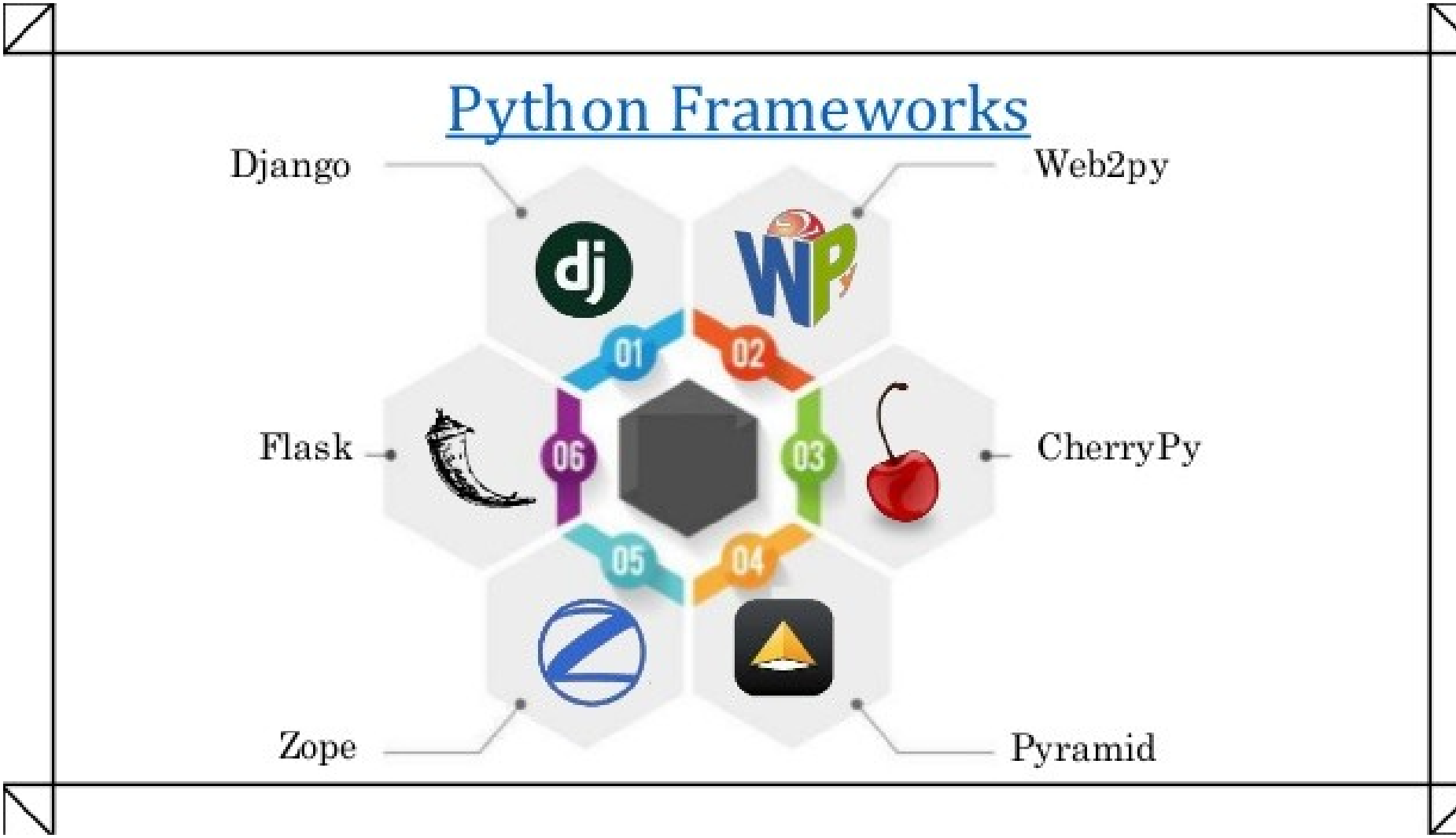
RESTful веб-сервисы

- Ресурс ориентированная технология
- HTTP запросы
- Для несложной бизнес-модели
- Работа с ресурсами, а не методами
- Легче разрабатывать

Пример вызова сервера погоды по SOAP (XML) протоколу

```
import requests
url="http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL"
#headers = {'content-type': 'application/soap+xml'}
headers = {'content-type': 'text/xml'}
body = """<?xml version="1.0" encoding="UTF-8"?>
          <SOAP-ENV:Envelope
xmlns:ns0="http://ws.cdyne.com/WeatherWS/"
xmlns:ns1="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
              <SOAP-ENV:Header/>

<ns1:Body><ns0:GetWeatherInformation/></ns1:Body>
          </SOAP-ENV:Envelope>"""

response = requests.post(url,data=body,headers=headers)
print response.content
```

Выбор микрофреймворка



REST on FLASK

https://flask.palletsprojects.com/en/2.1.x/#user-s-guide

```
# A Minimal Application
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Запуск приложения

BASH

```
$ export FLASK_APP=hello
$ flask run
* Running on http://127.0.0.1:5000/
```

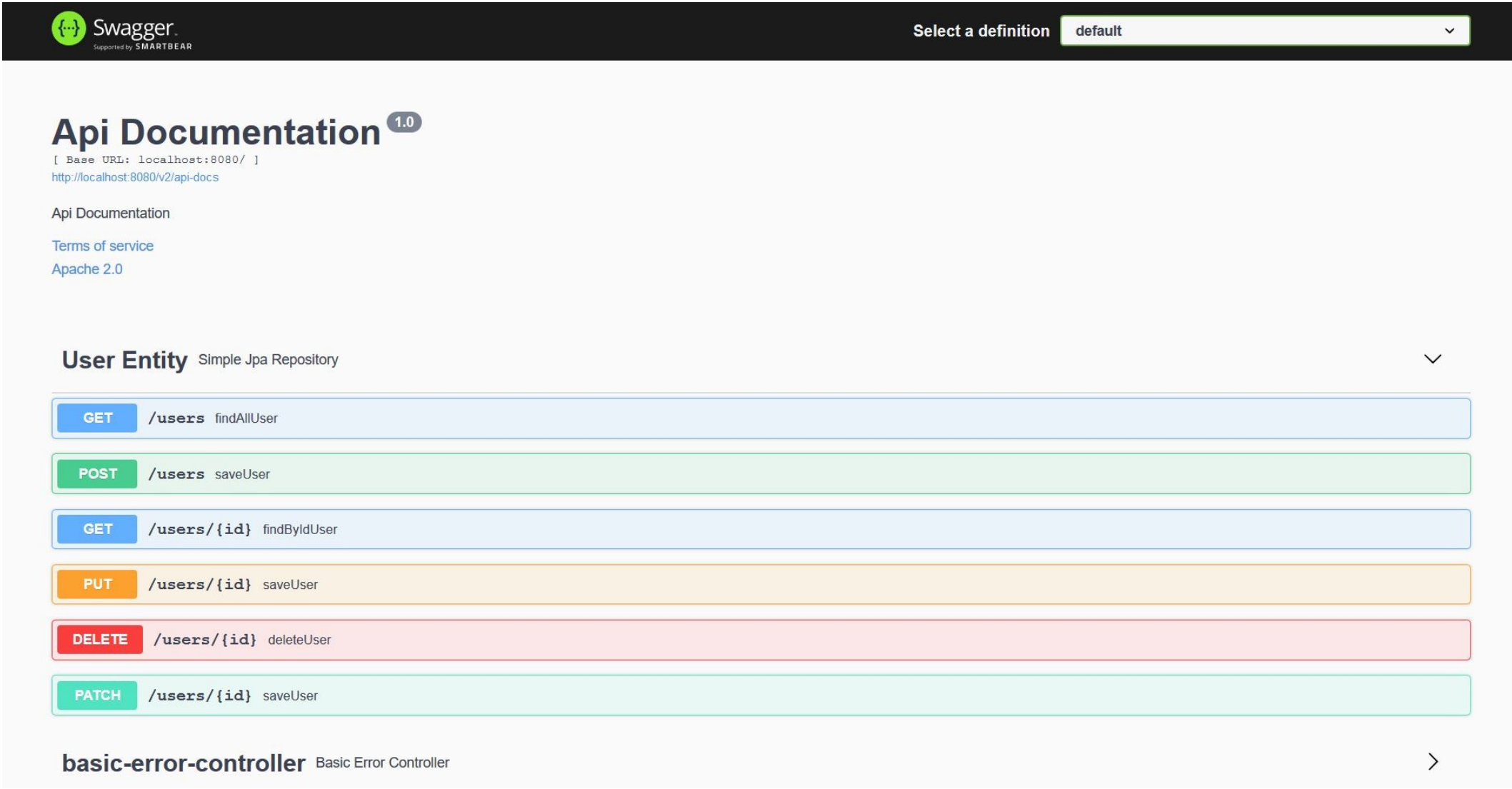
Маршруты (Routing)

```
from markupsafe import escape

@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return f'User {escape(username)}'

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an
integer
    return f'Post {post_id}'

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    # show the subpath after /path/
    return f'Subpath {escape(subpath)}'
```

HTTP методы

```
from flask import request


@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

Swagger - REST документация



Базовая структура OPENAPI 3.0

```yaml
openapi: 3.0.0
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9

servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
  - url: http://staging-api.example.com
    description: Optional server description, e.g. Internal staging server for testing

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':    # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

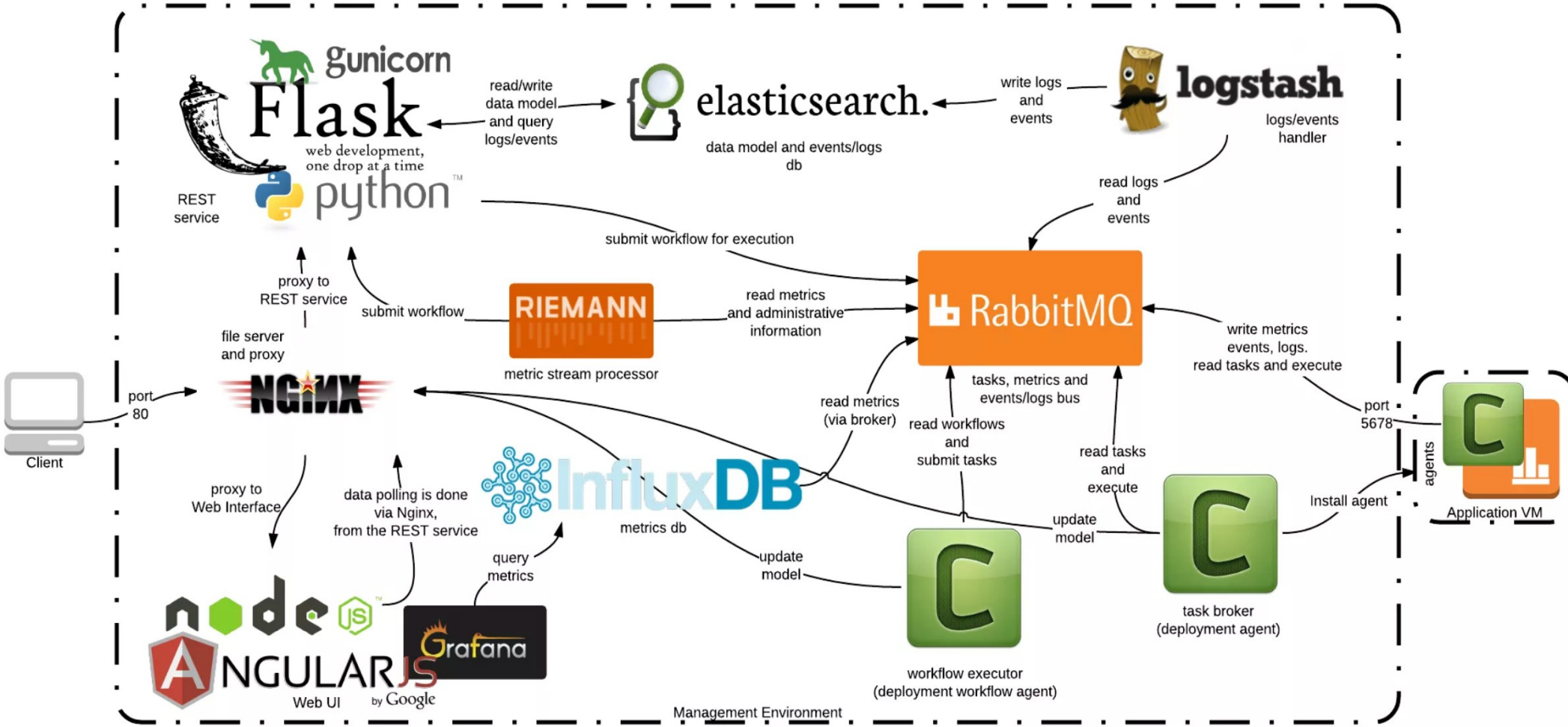
Мне нужно описывать свой REST в YAML ?

Flask API Documentation using Flask-Restx

https://flask-restx.readthedocs.io/en/latest/quickstart.html

```
#При установки и добалении библиотеки restx
#документация на REST API будет генерироваться
#автоматически.

from flask import Flask, request
from flask_restx import Api, Resource, reqparse


app = Flask(__name__)
api = Api(app)
@api.route('/hello')
class HelloWorld(Resource):
    def get(self):
        return 'hello'
if __name__ == '__main__':
    app.run()
```

Что из себя представляет web нового поколения?


gunicorn
Flask
web development, one drop at a time
python
REST service
read/write data model and query logs/events
elasticsearch.
data model and events/logs db
write logs and events
logstash
logs/events handler
read logs and events
submit workflow for execution
RabbitMQ
proxy to REST service
submit workflow
RIEMANN
metric stream processor
read metrics and administrative information
file server and proxy
NGINX
port 80
Client
tasks, metrics and events/logs bus
read metrics (via broker)
read workflows and submit tasks
write metrics events, logs. read tasks and execute
port 5678
agents
Application VM
read tasks and execute
Install agent
proxy to Web Interface
data polling is done via Nginx, from the REST service
query metrics
InfluxDB
metrics db
update model
update model
task broker (deployment agent)
workflow executor (deployment workflow agent)
node
ANGULARJS
by Google
Web UI
Grafana
Management Environment

Заключение

Web-сервисы — это новая технология интеграции Web-приложений. На текущий момент она используется в корпоративных приложениях для предоставления интерфеса взаимодействия с бизнес приложением.

Начните разрабатывать свой Web-сервис с уже имеющийся бизнес логики перенеся ее на выделенный сервис. Удачи!